Exercises in High-Dimensional Sampling: Maximal Poisson-disk Sampling and k-d Darts

Mohamed S. Ebeida, Scott A. Mitchell, Anjul Patney, Andrew A. Davidson, Stanley Tzeng, Muhammad A. Awad, Ahmed H. Mahmoud, and John D. Owens

Abstract We review our recent progress on efficient algorithms for generating well-spaced samples of high dimensional data, and for exploring and characterizing these data, the underlying domain, and functions over the domain. To our knowledge, these techniques have not yet been applied to computational topology, but the possible connections are worth considering. In particular, computational topology problems often have difficulty in scaling efficiently, and these sampling techniques have the potential to drastically reduce the size of the data over which these computational topology algorithms must operate. We summarize the definition of these sample distributions; algorithms for generating them in low, moderate, and high dimensions; and applications in mesh generation, rendering, motion planning and simulation.

1 Introduction

1.1 Maximal Poisson-disk Sampling (MPS) Definition

Sample points are called *well-spaced* if they have a limited ratio between the maximum distance between any domain point and its nearest sample point, and the minimum distance between two sample points. A well-spaced sampling is efficient at exploring a space. The maximum distance ensures that the domain is adequately covered and reduces interpolation error. The minimum distance ensures that we do

Mohamed S. Ebeida and Scott A. Mitchell
Computing Research, Sandia National Laboratories, e-mail: msebeid@sandia.gov

Anjul Patney, Andrew A. Davidson, Stanley Tzeng*, and John D. Owens
Dept. of Electrical and Computer Engineering (* Dept. of Computer Science), University of California, Davis, e-mail: jowens@ece.ucdavis.edu

Muhammad A. Awad, and Ahmed H. Mahmoud
Dept. of Naval Architecture and Marine Engineering, Alexandria University, Egypt

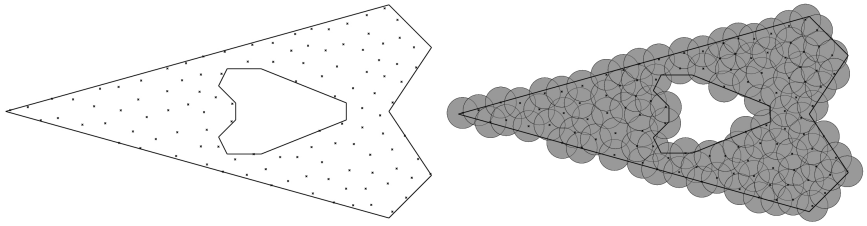


Fig. 1: A maximal Poisson disk sample over a non-convex domain and a uniform sizing function.

not waste time or generate noise with samples that provide similar information to nearby samples. Typically well-spaced is defined locally, by stating that the aspect ratio of Voronoi cells is bounded. Samples that have random positions are often preferred because they do not introduce directional bias in the estimates.

Maximal Poisson-disk Sampling (MPS) is a process that selects a random set of points, $X = \{x_i\}$, from a given domain, $\mathscr{D}$, in some d-dimensional space. The samples are at least a minimum distance apart, satisfying an empty disk criterion: Equation (2). For simplicity we focus on the uniform case, where the disk radius, r, is constant regardless of location or iteration. Inserting a new point, x_i, defines a smaller domain, $\mathscr{D}_i \subset \mathscr{D}$, available for future insertions, where $\mathscr{D}_o = \mathscr{D}$; see Equation (1). The *maximal condition*, Equation (3), requires that the sample disks overlap, in the sense that they cover the whole domain leaving no room to insert an additional point. This property identifies the termination criterion of the associated sampling process. *Bias-free* or *unbiased* means that the likelihood of the next sample being inside any remaining subdomain is proportional to the area of the subdomain; see Equation (1). This is uniform sampling from the uncovered area, equivalent to uniform sampling over the entire domain, and rejecting already-covered points. See Fig. 1 for an example MPS over a non-convex domain. Extending MPS disks to squares, or ellipses and rectangles in anisotropic spaces, is natural, yet unexplored.

$$\text{Bias-free: } \forall x_i \in X, \forall \Omega \subset \mathscr{D}_{i-1} : P(x_i \in \Omega) = \frac{\text{Area}(\Omega)}{\text{Area}(\mathscr{D}_{i-1})} \tag{1}$$

$$\text{Empty disk: } \forall x_i, x_j \in X, x_i \neq x_j : ||x_i - x_j|| \geq r \tag{2}$$

$$\text{Maximal: } \forall x \in \mathscr{D}, \exists x_i \in X : ||x - x_i|| < r \tag{3}$$

1.2 Applications of MPS

Random point distributions, including MPS, have found widespread use in computer graphics [17, 19]. Most applications are in dimensions below 6. Rendering Section (5.2) makes use of sampling light rays. The global illumination problem is concerned with computing indirect lighting, light from sources that are reflected off surfaces before illuminating an object in a scene. Computing the light in a scene exactly is intractable because of the number of combinations of light sources, ray directions, surface reflection angles, and observer positions. This is made worse if these quantities are unknown a priori, such as the location of an observing character in a game. An approximate solution makes use of MPS points. These points sample the scene space, and we can precompute the approximate contribution of each sample point, the inter-sample transmissions. Then, in real time, we may magnify by light source intensity and interpolate and combine with other information. A similar MPS sampling and workflow is done for texture synthesis. A small piece of a texture is placed at each MPS sample point, and the boundaries between patches are blended so as not to stand out to an observer.

In these graphics applications and some others, all three properties of MPS are desired: maximality ensures the accuracy of the sampled approximate solution; empty disk ensures efficiency by avoiding nearby, redundant points; and bias-free avoids artificial visual artifacts, repeating patterns, that a deterministic regular spacing produces. Humans are expert at detecting patterns, even imagining them where none exists. The distribution of retinal cells in our eyes has a Fourier spectrum much like that of MPS, which may help explain why MPS works so well.

MPS points are well-spaced, which leads to meshes with well-shaped elements. Other meshing algorithms generate well-spaced points, but MPS is also random. For simulating fracture, e.g. for carbon sequestration in Fig. 2, meshes with both properties are required [2, 7]. The mesh randomness models some of the natural material strength variability. Generating different meshes for the same geometry using the same sizing function provides a useful tool to study the sensitivity of the solution to the mesh, complementary to refinement studies; see Fig. 6(a).

1.3 Potential Ties to Computational Topology

We speculate that some MPS techniques and applications may be useful for analysis of data for computational topology. For example, as in global illumination, MPS could be used as cluster centers as a form of resampling the domain data, to reduce the size of a computational topology point cloud. MPS processes could also be used to generate the initial data points, perhaps modified to place more points where they are topologically significant. In particular, many computational topology algorithms rely on a mesh. Our sampling techniques can be used to create such a mesh. Going further, we may circumvent the need for a mesh by using the structure implied by the

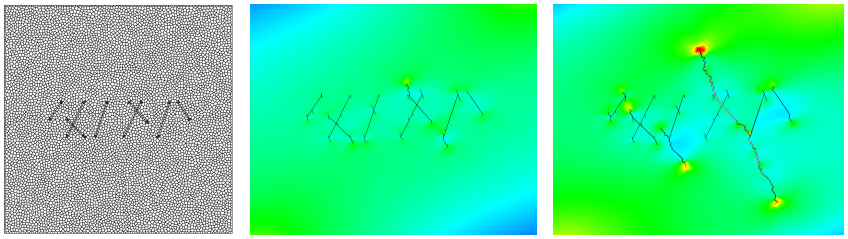
disks. For example, we are investigating implicit Voronoi diagrams for traversing a point cloud to construct the Morse-Smale complex.

One recent use of MPS points in high dimensions is real-time robot motion planning, where the space is the configuration space of the robot and its obstacles [18]. The challenges for motion planning are similar to those for some discrete computational topology calculations. Both rely on an imperfect representation of the space by point clouds. Algorithms for both problems typically suffer from the curse of dimensionality, and the point clouds are typically high dimensional. In motion planning the goal is to find one path between two points, which has similarities to finding Morse-Smale or Reeb graph paths, and contrasts to trying to characterize the entire space in homology calculations. Recently a realtime robot motion planning problem was solved using our MPS sampling of its 23-dimensional configuration space, so there is hope that large spaces from topology might become computationally tractable as well. Conversely, it is possible that computational topology techniques for finding smooth and short paths, e.g. homology generators and Morse-Smale paths, could be used to smooth and shorten a robot path. Smooth and short paths mimic human motions and are more efficient.

2 MPS Algorithms in Low Dimensions

2.1 Algorithmic Challenges

Poisson-disk sampling is defined as a serial statistical process of rejection sampling: generate a disk uniformly at random and reject it if its center lies inside a prior disk. A maximal sampling is defined as achieving the limit distribution. A straightforward implementation of the statistical process is called "dart throwing," where a dart is synonymous with a candidate disk center point. In dart throwing, most darts are



(a) Initial Voronoi mesh. (b) Early time. (c) Late time.

Fig. 2: Fracture after injecting CO2 below caprock. The color represents maximum principal stress. The initial fracture joints are sealed, but opened and spread due to the high injection pressure.

accepted in the beginning of the process, but the likelihood of accepting the next point is proportional to the volume fraction of the domain left uncovered by a disk, which typically becomes vanishingly small as the process continues.

A process definition is not the same as an output definition, nor necessarily the most efficient way of achieving that output. (Consider defining "sorted order" as the output of the bubble sort process. Only after "sorted order" is defined independent of the process that produced it do we have the chance to discover quicksort.) Unfortunately we (the community) do not have a precise closed-form description of the MPS output distribution, nor are we sure that it is the ideal for random well-spaced points [14]. One branch of research has modified the MPS process in order to achieve efficiency [5]. Our (the authors') first solutions are in a different research branch, where we propose algorithms that produce equivalent outputs to dart throwing, but more efficiently.

2.2 Our Algorithmic Solutions

In particular, to find an efficient equivalent process, we rely on the observation that the probability of introducing the next disk center in any uncovered subregion is proportional to the area of the subregion. We use a background grid subdivision of the domain to track a superset of the remaining uncovered subregions. Cells that are completely covered by a single disk are discarded. Efficiency follows from the superset being not much bigger than the uncovered region. The background grid is uniform with cell diagonals of length r. This size ensures a cell can have at most one point, whose disk completely covers the cell. The background grid also speeds up retrieving nearby disks, to check if a sample point is inside one. We now summarize two variations that use this grid [11, 13].

2.2.1 Efficient MPS by Polygon Tracking

The first MPS algorithm we developed has two phases [13]. The first phase is dart throwing, but each dart is selected within a grid cell, not the entire domain. After a

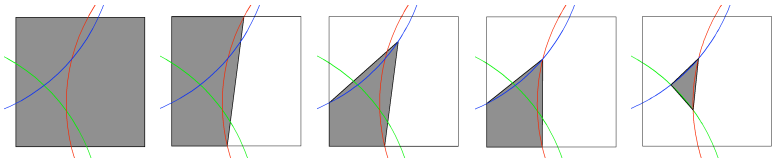


Fig. 3: Generating a tight polygonal superset of the uncovered regions. From left to right, we start with the cell, then subtract disks. We use the chords instead of the arcs between intersection vertices.

number of dart throws proportional to the number cells, we switch to the second phase. We further refine each cell by constructing a polygon that is closer to the uncovered region inside the cell, but is still a superset; see Fig. 3. Now we do dart throwing inside the polygons: we select a polygon uniformly by area, then a dart uniformly inside it. If the dart is uncovered we accept it and update the polygons. The chance of a dart being uncovered is provably large, which leads to a provable expected run-time of $\mathbb{E}(n \log n)$. (There is typically no deterministic time bound, e.g. $O(n \log n)$, for these types of algorithms because of the random decisions they must make.) Unbeknownst to us there was a prior method with the same guarantees of maximality, bias-free, and $\mathbb{E}(n \log n)$ time [15]. It uses the evolving Delaunay triangulation to keep track of the remaining void. Ours [13] was the first method based on grids with these guarantees; grids are preferred in some settings due to their locality, simplicity, and speed. Our method was easily parallelized on a GPU. However, keeping track of polygon intersections is both cumbersome and requires a large amount of memory. If we increase the domain dimension the memory consumption explodes, restricting this method to low-dimensional spaces. The next method we developed addresses these shortcomings for slightly higher dimensions, and we prefer it even for lower dimensions.

2.2.2 Simple MPS by Implicit Quad-Trees

The second method [11] we developed, Simple MPS, maintains all of the desirable qualities of our previous method, Efficient MPS [13], namely maximality and bias-free, with the added benefits of being simpler to code; using less memory; faster run-time even in low dimensions and scalable to higher dimensions in practice (but without a run-time proof). It starts with dart throwing in the background grid; described in Section (2.2.1) and identical to the first algorithm [13]. However, instead of proceeding to polygons, we simply subdivide all the remaining child cells. Covered child cells are discarded, and we repeat the algorithm on the remaining cells. Since the cells are all the same size, it is easy to represent them by indices, and we do not need the overhead of a tree as in a true quadtree.

The key to this strategy's efficiency is that the *collection* of active cells is a close approximation to the *entire* uncovered area, even if one particular cell is much larger than the uncovered area it encloses. In practice the number of cells decreases geometrically per refinement, which helps both runtime and memory, and allows us to reach maximality by refining down to machine precision. We were able to maximally sample 6d domains on a CPU. On the GPU, we sampled at an impressive rate of 1M samples/sec in 2d and 75K samples/sec in 3d. More details on how to parallelize, and proofs of maximality and bias-free, can be found in our paper [11].

2.3 Variable Radii MPS

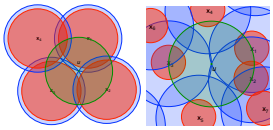
We define two useful variations of MPS [16]. In the first version, the size of the disks varies spatially over the domain; see Fig. 4(b) for a classic stippling application. We still get locally well-spaced points if this variation is slow, e.g. the disk sizing function satisfies a Lipschitz condition. The quality of the meshes generated from those points degrades smoothly as the variation increases, up to some critical threshold after which there are no guarantees. The run-time also increases with the variation, because proximity checks must search a larger neighborhood. Spatially varying radii had been considered previously by other authors [3], but we appear to be the first to quantify these conditions and their effects [16].

If the radii of two disks differ, there are several ways of defining "conflict," the conditions under which a dart is rejected because it fails to satisfy a version of Equation (2). There are variations based on size and generation order. Each variation has advantages and disadvantages. Defining conflict as the smaller disk containing the center of the larger disk provides the best quality and it can tolerate the largest Lipschitz constant (< 1), but generates the largest point sets. If we define a conflict as a candidate disk center lying inside a previously accepted disk, then this is the easiest to implement, as it is a minor change to Simple MPS and other algorithms. However, it has the biggest restriction on the Lipschitz constant ($< 1/2$) and provides the weakest output quality guarantees. We have also explored *sifted disks* for reducing the discrete density of a maximal point set, by removing and relocating points, while still maintaining the MPS conditions [8].

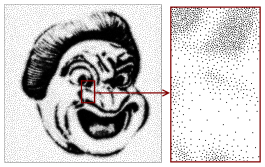
Our second useful variation of MPS [16] is to use two radii for each point, two-radii MPS; see the concentric blue and red disks in Fig. 4(a). We decouple the maximality (coverage, blue) and conflict (empty-disk, red) conditions, by using different disk radii for each. I.e. we replace the disk-free "r" in Equation (2) with r_f, and replace the coverage "r" in Equation (3) with r_c. One benefit of a *larger* coverage radius r_c is a smoother noise spectrum [20], defined by the Fourier transform of all pairwise point distances; see Fig. 5. We have also explored adapting a point set to obtain a *smaller* coverage radius, which improves interpolation error and mesh quality, by a method we call opt-beta, locally optimizing the position of nodes [6].

3 Meshing Algorithms Based on MPS

We consider two types of meshes: constrained Delaunay triangulations [10] and Voronoi polyhedra [9]. See Fig. 6 for triangles and Fig. 8 for poyhedra. MPS produces well-spaced points, which can lead to well-shaped elements immediately, without the need for post-processing such as smoothing or edge swapping; see Fig. 7(b). For Delaunay triangulations we must place points exactly on the domain boundary to get a conforming mesh. We must place boundary points more densely than in the interior to ensure good quality triangles. In Voronoi meshing, the MPS points are the Voronoi cell seeds, and it is better if the points lie strictly interior to the domain, and the cells

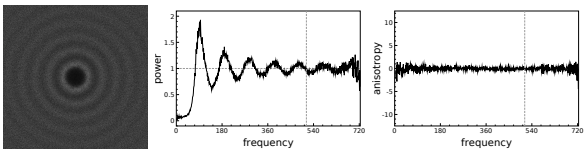


(a) Two-radii, $r_c > r_f$.

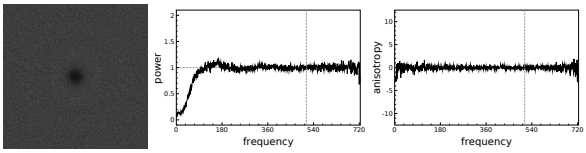


(b) Spatially varying radii.

Fig. 4: Variable radii MPS.

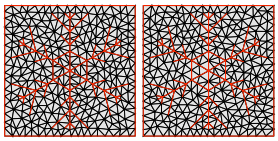


(a) Single radius MPS

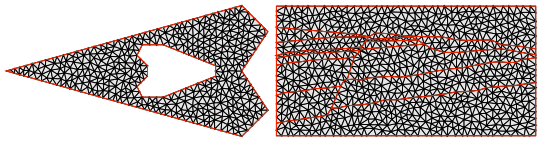


(b) Two radii, $r_c = 2r_f$.

Fig. 5: Spectral properties [20] of single-radii and two-radii MPS.



(a) Two random meshes with the same radius and domain.

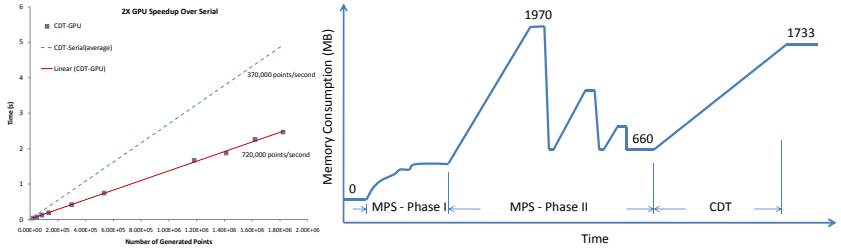


(b) Random meshes of non-convex domains. Red internal interfaces are represented in the mesh.

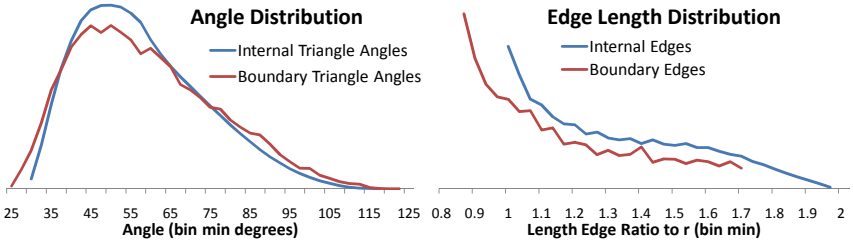
Fig. 6: Delaunay triangulations via Poisson-disk sampling.

are clipped by the domain boundary. Using interior points, cells have better aspect ratios, and the Voronoi mesh is smaller.

Delaunay refinement [4] is the most popular method for well-spaced points and well-shaped triangles. It creates triangles first, and adds points to remove bad-quality triangles. In contrast, MPS generates well-spaced points first, and only forms triangles at the end. The theoretical guarantees about the outputs, the numbers of points and elements' qualities, are nearly identical. In practice, freedom from the vagaries of intermediate triangles appears to allow MPS to change mesh size more quickly [1], and more closely adhere to a sizing function, for the same quality requirements. The local nature of MPS operations leads to near-linear time in serial; see Fig. 7. It also leads to easy parallelism with little communication [10]. In contrast Delaunay refinement must build very large intermediate triangles between distant points.



(a) Near-linear run-time (left) and memory (right).



(b) Provably-good angle and edge length quality.

Fig. 7: Algorithm performance and mesh quality for our MPS triangle mesher.

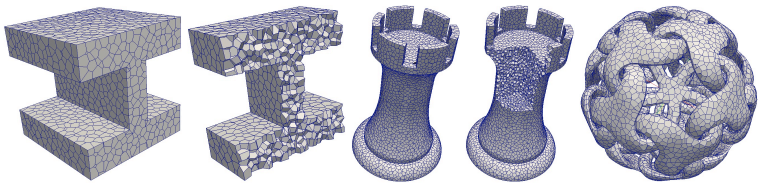


Fig. 8: MPS Voronoi meshes. The boundary elements are nearly as large and well-shaped as the interior elements. Operations are local so it is easy to handle complicated global topology.

4 Sampling in High Dimensions

4.1 Algorithmic Challenges

Although the definition of MPS is dimension-independent, some properties of the distribution change as the dimension increases, and the algorithms that work well in low dimensions take more memory and time. High-dimensional spaces present several challenges. As dimension increases, the volume of a sphere relative to its bounding box decreases, and relative to its inscribed box it increases. A unit box can contain an exponential-in-d number of unit disks. The box becomes a worse

approximation of a sphere, and this dooms grid-based methods as expensive in time, memory, or both. Regardless of the methods used to generate the sampling, each disk can have exponentially more nearby disks as the dimension increases. This increases the combinatorial complexity of computing intersections of disks, or disks with grids.

If we take a step back from MPS, a typical underlying goal is to sample the space in a way that gives unbiased estimates, with low variance, of some quantity. Typically this still means our sampling should be random. However, points are just one way of sampling, a zero-dimensional way. MPS points are chosen uniformly at random *by volume* without regard to the shape of the domain. Thus it is hard to hit narrow regions with point samples, even though some of their dimensions might be large. In particular, in standard MPS algorithms it is hard to tell if maximality has been reached and the domain is covered by disks. More generally, it is hard to track narrow regions of interest. In uncertainty quantification, the domain may be the parameter space of a simulation, and the region of interest is where the simulation returns a value below a threshold. Typically the simulation is more sensitive to some parameters than others, so this subregion is narrow in those dimensions, but large in the insensitive parameter directions.

4.2 Algorithmic Solution: k-d Darts

To address such a scenario, rather than evaluating a function at a single point, we consider higher-dimensional evaluations along k-dimensional hyperplanes or "flats" [12]. Initially, we defined a "k-d dart" as a set of axis-aligned hyperplanes spanning all combinations of k free and $d-k$ fixed coordinates. However, further analysis and experimentation showed that just picking k-dimensional hyperplanes with random free-coordinate indices is simpler and works just as well. This is all

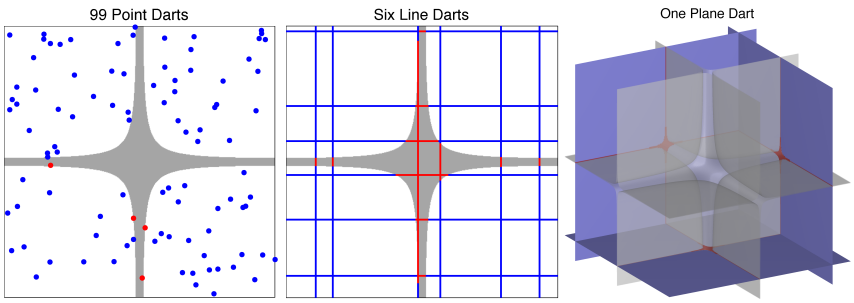


Fig. 9: Sampling a narrow region with hyperplanes increases the amount of information we capture. While the number of random points landing in the gray area approaches zero as its thickness decreases, each random line is destined to capture a portion of it.

predicated on the assumption that it is possible to evaluate a function along a flat. A particularly nice situation is when the function is analytic, and we can substitute the flat's fixed coordinates for its parameters. As a last resort, one could estimate the evaluation of a function along a flat numerically. In general the cost of evaluating a higher-dimensional flat is typically larger than a 0-d flat (i.e. a point). However, the amount of information gained and the ability to find narrow regions is often worth the cost. We get faster convergence, and higher quality in our examples.

5 High-Dimensional Algorithms using k-d Darts

We next show the utility of darts over several algorithmic examples: generating relaxed maximal Poisson-disk samples, approximating depth of field blur [12, 21], and volume estimation [12].

5.1 Relaxed Maximal Poisson-disk Sampling (RMPS)

In a maximal Poisson-disk sample, the coverage disks overlap to cover the entire domain, leaving no room to increase the sample size. At maximality, the achieved coverage radius r_c is at most the prescribed conflict radius r_f; recall Section (2.3). Achieving this condition is extremely hard in high dimensions (e.g. $d > 6$). A relaxed version of this condition allows the coverage disks to be slightly larger than the conflict disks, quantified by their ratio, the *distribution aspect ratio* $\beta = \frac{r_c}{r_f} > 1.0$. Increasing the allowable upper bound on β makes solving the problem easier. k-d darts [12] utilize line darts to capture the narrow voids between existing conflict disks while simultaneously estimating the volume of the remaining void. Line darts are much faster than point darts for RMPS, because it is fast and easy to subtract a set of spheres from an axis-aligned line. The remaining segments are uncovered, and we introduce a new sample along them.

Our k-d darts RMPS method produces nearly-maximal distributions whose spectral properties (randomness) are nearly identical to those of MPS. It uses much less memory than previous methods, allowing us to examine larger and higher-dimensional domains. We sample domains of up to 6 dimensions on a CPU, and 3 dimensions on a GPU.

5.2 Depth of Field by Line Darts (DoF)

Optical camera lenses focus at a single distance from the camera, so captured images typically exhibit blur effects at other distances. Computer-generated images, on the other hand, by default are in focus at all distances. Simulating the depth-of-field

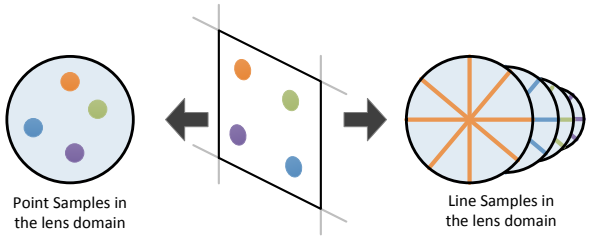


Fig. 10: Depth of field line sampling vs. point sampling. We have four colored samples in scene space (middle). For each one, point sampling generates one point in lens space (left), while our method generates multiple line samples (right).

effects of a real camera is helpful in adding realism to computer graphics. Using one-dimensional darts or line darts or line samples [12, 21], one can produce high-quality low-noise images with depth-of-field effects. Compared to traditional point sampling, line darts are able to capture more information per sample. Although each line dart is computationally more expensive than a point dart, in practice line-darts reduce the overall run time to produce an image of comparable quality.

Depth of field involves sampling the image in 4-d (x,y,u,v) space, where (x,y) is screen space and (u,v) is lens space. In k-d darts [12], each dart sample consists of four orthogonal lines, one spanning each of the 4-d coordinates. In *wagon wheel* [21], each dart sample consists of a radial line in (u,v) space, passing through the center of the lens; see Fig. 10. The remainder of this section elaborates on the wagon wheel approach.

For each pixel we generate several sampling locations (x_p,y_p) within the pixel and then perform line sampling along the lens (u,v) space. Rendering starts by computing intersections between line darts and incoming primitives that represent the scene. Primitives that intersect the line samples then generate colored line segments, whose contribution is aggregated in to the final color of the pixel. How line samples are placed along the lens plays a crucial role in the final image quality.

A wagon wheel line sampling pattern (Fig. 10) has several advantages when compared to alternatives. First, it has uniform line sample lengths. This helps when implementing the algorithm on highly parallel architectures, such as a GPU. Second, line samples passing through the origin can be expressed in a simple slope-intercept form $v = mu$, simplifying the math. Lastly, the bias that is associated with such a pattern can be easily removed by a reweighting during filtering. An alternative would be a grid-like pattern, as the ridges of a waffle, using k-d darts in just (u,v) space.

Figure 11 shows the results of several scenes rendered with 256 point samples versus 16 wagon wheel line samples, run in parallel on a NVIDIA GTX 580 GPU. Also, for k-d darts, 16 line-darts produce a better picture, more quickly, than 1024 points. Using either wagon wheels or k-d darts, line sampling demonstrates a clear win in terms of both quality and performance.

(a) Example 800 × 800 pixel scenes rendered using 256 *point* samples.

(b) Example 800 × 800 pixel scenes rendered using 16 *line* samples.

Fig. 11: 256 point samples (top) and 16 wagon-wheel line samples (bottom) per pixel produce nearly the same image, but line samples are about 4 times faster.

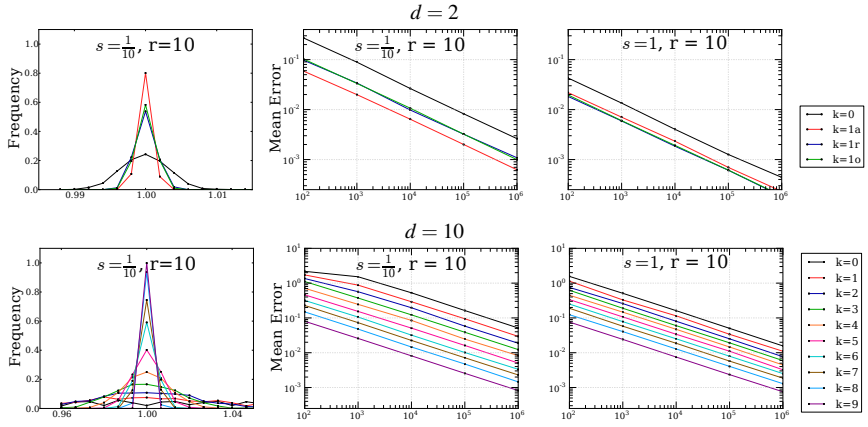


Fig. 12: d-dimensional hyper-ellipsoid volume estimation, for varying squish factor s scaling the main axis, dart dimension k, and fixed number of random rotations $r = 10$ of the ellipsoid. Top uses axis-aligned (1a), randomly oriented (1r), and orthogonal pairs of randomly oriented darts (1o). Bottom uses axis-aligned darts in 10-d. Left shows the ratio of the estimated to true volume by frequency for a fixed number of samples, n. Right shows $|\text{mean} - \text{true}|/\text{true volume}$ by the the number of darts, n.

5.3 Volume Estimation by Hyperplane Darts

We study the accuracy of high-dimensional sampling using k-d darts, on the classical problem of estimating the volume of an object [12]. We seek to experimentally

quantify the effects of the object orientation, dart orientation, object surface area to volume ratio, and dimensions of the object and dart. We create a d-dimensional hyper-ellipsoid object as follows.

- Start with a unit d-ball (d-dimensional unit sphere);
- Scale along the x-axis by a factor squish s to generate an elliptical profile; and
- Perform r random Givens rotations to randomly orient it.

We compute the volume of the ellipsoid analytically, for the ground-truth. For comparison, we estimate the volume of the ellipsoid using $k = 0$ darts, i.e. classical Monte Carlo point sampling: sample random points from the ellipsoid's bounding box and count the fraction inside the ellipsoid. The accuracy decreases as d increases.

For k-d darts, we can choose to throw line darts, plane darts, or darts of any dimension $k < d$. Fig. 12 show how k-d darts consistently outperform point darts. Their advantage increases as k increases.

Darts may be axis-aligned or arbitrarily oriented. We recommend axis-aligned darts for three reasons. First, it is easy to distribute aligned darts uniformly, which ensures that the expected mean of the function estimates is accurate. Second, it is easiest to implement aligned darts, since it involves simply fixing coordinate values. Third, in many cases it is most efficient because we may obtain an expression for the underlying function along a dart by substituting in the fixed coordinate values. We compare the accuracy of aligned and unaligned darts in the top row of Fig. 12. For squished ellipsoids, aligned darts are slightly more accurate, but the prior reasons are more significant.

6 Summary

We have highlighted the main research results of our sampling group from 2011 to 2013. We consider MPS, or more generally point sets that are both well-spaced and random, to be a very rich area, crossing many fields and applications. Over the course of our research, we have explored many modifications to our algorithms and used the output for many different applications. Uses and features of our sample generation and modification algorithms are summarized in Tables 1 and 2. We have compared and contrasted these to the works of others. These variations [6, 8], and their tradeoffs, led us to the conceptual framework for sampling illustrated in Fig. 13.

Acknowledgements The UC Davis authors thank the National Science Foundation (grant # CCF-1017399), Sandia LDRD award #13-0144, UC Lab Fees Research Program Award #12-LR-238449, NVIDIA and Intel Graduate Fellowships, and the Intel Science and Technology Center for Visual Computing for supporting this work.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

name	goals	mesh	nodes	beta	dim
Opt-beta	reduce beta	ok	move	tune < 1	2
Steiner reduction	fewer points, preserve mesh angles	yes	(re)move	≥ 1	2
Sifted disks	fewer points, preserving beta=1	ok	(re)move	1	2
k-d darts	create points, integration, in high dimensions		create	$\rightarrow 1$	2–23
Variable radii MPS	create points with sizing function or spacing	yes	create	tune ≥ 1	2
VorMesh MPS	create Voronoi polyhedral mesh	yes	create	1–2	2–3
DelMesh MPS	create Delaunay triangle mesh	yes	create	1–2	2–3
Simple MPS	create points in moderate dimensions	ok	create	1	2–6
Efficient MPS	create points in two dimensions	ok	create	1	2

Table 1: Our sample generation and modification algorithms' application context.

name	GPU	time	memory	features	ref
Opt-beta		numeric	n	local position optimization, as smoothing	[6]
Steiner reduction		$\approx n$	n	2-to-1 node replacement	[1]
Sifted disks		n	n	2-to-1 node replacement	[8]
k-d darts		n^2d^2	nd	high-d, global hyperplanes, many applications	[12]
Variable radii MPS		$\approx n$	n	provable qualities	[16]
VorMesh MPS		$\approx n$	n	bounded domains, provable quality	[9]
DelMesh MPS	GPU	$\approx n$	n	bounded domains, provable quality	[10]
Simple MPS	GPU	$\approx n2^d$	$n2^d$	efficient flat quadtree tracking voids	[11]
Efficient MPS	GPU	$n \log n$	n	provable runtime, polygon approx. voids	[13]

Table 2: Our algorithms' performance and features. Many [9, 10, 16] use a form of Simple MPS [11].

References

1. Abdelkader, A., Mitchell, S.A., Ebeida, M.S.: Steiner point reduction in planar Delaunay meshes. In: ACM Symposium on Computational Geometry, p. to appear (2014)
2. Bolander Jr., J., Saito, S.: Fracture analyses using spring networks with random geometry. Engineering Fracture Mechanics **61**(56), 569 – 591 (1998). DOI http://dx.doi.org/10.1016/S0013-7944(98)00069-1. URL http://www.sciencedirect.com/science/article/pii/S0013794498000691
3. Bowers, J., Wang, R., Wei, L.Y., Maletz, D.: Parallel Poisson disk sampling with spectrum analysis on surfaces. In: SIGGRAPH Asia '10, pp. 166:1–10 (2010)
4. Chew, L.P.: Guaranteed-quality mesh generation for curved surfaces. In: Proceedings of the Ninth Annual Symposium on Computational Geometry, pp. 274–280 (1993)
5. Dunbar, D., Humphreys, G.: A spatial data structure for fast Poisson-disk sample generation. ACM Trans. Graph. **25**(3), 503–508 (2006). DOI 10.1145/1141911.1141915. URL http://doi.acm.org/10.1145/1141911.1141915
6. Ebeida, M.S., Awad, M.A., Ge, X., Mahmoud, A.H., Mitchell, S.A., Knupp, P.M., Wei, L.Y.: Improving spatial coverage while preserving the blue noise of point sets. Computer-Aided Design (2013). DOI http://dx.doi.org/10.1016/j.cad.2013.08.015. In press.
7. Ebeida, M.S., Knupp, P.M., Leung, V.J., Bishop, J.E., Martinez, M.J.: Mesh generation for modeling and simulation of carbon sequestration process. In: DOE Scientific Discovery through Advanced Computing (SciDAC) (2011)

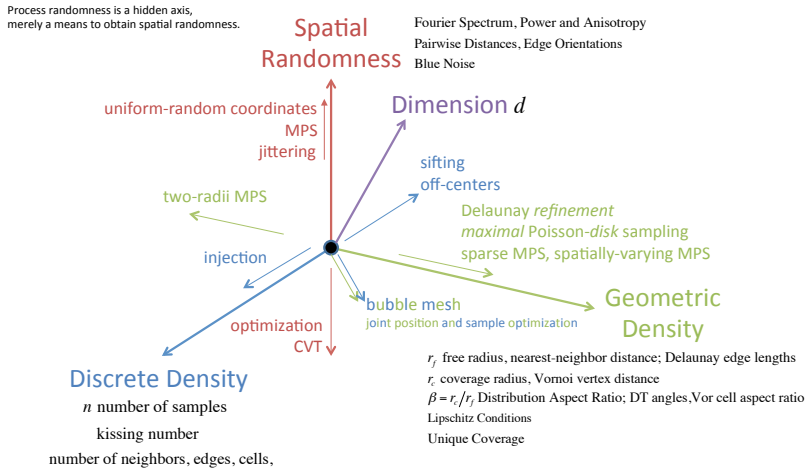


Fig. 13: A conceptual space parameterizing the output of any sampling method. Note methods such as jittering add randomness, while optimization methods tend to remove randomness. Each axis may be varied, but the axes are not independent. For example, two-radii MPS places points further apart, and hence produces fewer of them, and their positions are more random. Achieving maximality tends to add regularity and improve mesh quality. Bubble mesh interleaves changing the position and number of points; again optimizing positions leads to greater regularity.

8. Ebeida, M.S., Mahmoud, A.H., Awad, M.A., Mohammed, M.A., Mitchell, S.A., Rand, A., Owens, J.D.: Sifted disks. Computer Graphics Forum **32**(2pt4), 509–518 (2013). DOI 10.1111/cgf.12071
9. Ebeida, M.S., Mitchell, S.A.: Uniform random Voronoi meshes. In: 20th International Meshing Roundtable, pp. 273–290. Springer Berlin Heidelberg (2011). DOI 10.1007/978-3-642-24734-7_15
10. Ebeida, M.S., Mitchell, S.A., Davidson, A.A., Patney, A., Knupp, P.M., Owens, J.D.: Efficient and good Delaunay meshes from random points. Computer Aided Design **43**(11), 1506–1515 (2011). DOI 10.1016/j.cad.2011.08.012
11. Ebeida, M.S., Mitchell, S.A., Patney, A., Davidson, A.A., Owens, J.D.: A simple algorithm for maximal Poisson-disk sampling in high dimensions. Computer Graphics Forum **31**(2), 785–794 (2012). DOI 10.1111/j.1467-8659.2012.03059.x
12. Ebeida, M.S., Patney, A., Mitchell, S.A., Dalbey, K.R., Davidson, A.A., Owens, J.D.: *k*-d darts: Sampling by *k*-dimensional flat searches. ACM Transactions on Graphics (2014). In press. Also at arXiv:1302.3917 [cs.GR]; URL http://arxiv.org/abs/1302.3917
13. Ebeida, M.S., Patney, A., Mitchell, S.A., Davidson, A., Knupp, P.M., Owens, J.D.: Efficient maximal Poisson-disk sampling. ACM Transactions on Graphics **30**(4), 49:1–49:12 (2011). DOI 10.1145/1964921.1964944
14. Heck, D., Schlömer, T., Deussen, O.: Blue noise sampling with controlled aliasing. ACM Transactions on Graphics (TOG) **32**(3), 25 (2013)
15. Jones, T.R.: Efficient generation of Poisson-disk sampling patterns. Journal of graphics tools **11**(2), 27–36 (2006)
16. Mitchell, S.A., Rand, A., Ebeida, M.S., Bajaj, C.: Variable radii Poisson-disk sampling. In: Canadian Conference on Computational Geometry, vol. 24, pp. 185–190 (2012)

17. Nehab, D., Hoppe, H.: A fresh look at generalized sampling. Foundations and Trends in Computer Graphics and Vision **8**(1), 1–84 (2014). DOI 10.1561/0600000053. URL http://dx.doi.org/10.1561/0600000053
18. Park, C., Pan, J., Manocha, D.: RealTime GPU-based motion planning for task executions. In: IEEE International Conference on Robotics and Automation Workshop on Combining Task and Motion Planning (2013)
19. Pharr, M., Humphreys, G.: Physically based rendering: From theory to implementation. Morgan Kaufmann (2010)
20. Schlömer, T.: PSA point set analysis. Version 0.2.2. http://code.google.com/p/psa/ (2011)
21. Tzeng, S., Patney, A., Davidson, A., Ebeida, M.S., Mitchell, S.A., Owens, J.D.: High-quality parallel depth-of-field using line samples. In: Proceedings of High Performance Graphics, pp. 23–31 (2012). DOI 10.2312/EGGH/HPG12/023-031